Microcomputer, Electronic Equipment, and Debugging System

Technical Field

The present invention relates to a microcomputer and also to electronic equipment and a debugging system comprising the same.

Background of Art

There has recently been increasing demand for the incorporation of microcomputers that are capable of implementing high-level information processing into electronic equipment such as game machines, car navigation systems, printers, and portable information terminals. Such a thus-incorporated microcomputer is usually mounted on a user board called a target system. A software development support tool called an in-circuit emulator (ICE) is widely used for supporting the development of software to be used in the target system.

The CPU-switching type of ICE shown in Fig. 1A is the most common type of this kind of ICE used in the art. With this CPU-switching ICE, a microcomputer 302 is removed from a target system 300 during debugging, and a probe 306 of a debugging tool 304 is connected thereto instead. This debugging tool 304 emulates the operation of the removed microcomputer 302. The debugging tool 304 can also perform various processes necessary for debugging.

However, this CPU-switching ICE has disadvantages in that there is a large number of pins on the probe 306 and cables 308 of the probe 306. It is therefore difficult to emulate the operation of the microcomputer 302 at high frequencies (the limit is at approximately 33 MHz, by way of example). It is also difficult to

design the target system 300. Furthermore, the operating environment of the target system 300 (signal timings and load conditions) changes between when the microcomputer 302 is installed and is operating as designed and when in debugging mode when the debugging tool 304 is emulating the operation of the microcomputer 302. This CPU-switching ICE also has problems in that, if a different microcomputer is used, even if it is a modified version thereof, it is necessary to use a debugging tool of a different design and a probe in which the numbers and positions of the pins are different.

A known method of solving these disadvantages of such the CPU-switching ICE is an ICE in which debugging pins and functions for implementing the same functions as those of the ICE are installed on a mass-produced chip. This type of ICE with mounted debugging functions usually has a user mode and a debugging mode. A user program is executed in user mode and a debugging program is executed in debugging mode.

With a microcomputer having a user mode and a debugging mode, it can happen during debugging that a user program runs away or gets stuck in an infinite loop while it is executing. In such a case, means for forcibly switching from user mode to debugging mode is necessary. For that reason, a forced break function is provided, and a dedicated external terminal for implementing this forced break is usually also provided.

If an external input terminal for implementing such a forced break is provided, the number of pins on the package increases. It is, however, preferable to have as few terminals as possible that are necessary only for debugging and are not necessary for the end user.

It is also essential with this type of ICE with mounted debugging functions to communicate with external components (such as hardware other than the chip). However, communications with an external debugging tool are used only in debugging mode. This means that such communications terminals are not used in user mode.

Since terminals for inputting forced breaks and terminals used only in debugging mode are unnecessary for the end user, it is preferable to have as few of them as possible.

Disclosure of Invention

The present invention was devised in the light of the above technical concerns and has as an objective thereof the provision of a microcomputer that has a reduced number of terminals that are unnecessary to the end user, such as a terminals for inputting forced breaks and terminals used only in a debugging mode, in a type of ICE that has debugging pins and functions mounted on a mass-produced chip, together with electronic equipment and a debugging system that comprises the same.

To solve the above described technical problems, the present invention relates to a microcomputer having a user mode and a debugging mode, the microcomputer comprising: a central processing unit formed to be switchable between the user mode and the debugging mode, for executing instructions in each of the user mode and the debugging mode; and switching means for switching the central processing unit from the user mode to the debugging mode when a forced break is input through a terminal that is not used in the user mode.

A forced break in this context forcibly causes a transition from the user mode to the debugging mode.

In this aspect of the invention, a terminal that is not used in the user mode could be a terminal that is used only in the debugging mode for inputting a forced break, by way of example. Since it is therefore not necessary to provide a dedicated terminal for forced break, the number of terminals of the microcomputer can be reduced and it is possible to ensure that a larger number of terminals can be utilized by the user.

In another aspect of the present invention, the microcomputer has an on-chip debugging function and comprises a debugging terminal connected to a communications line for transferring debugging information, that is used for on-chip debugging, to and from an external debugging tool; and a forced break is input through the debugging terminal.

With a microcomputer having an on-chip debugging function, it is usually necessary to communicate with external components (hardware other than the chip). However, data is transferred to and from an external debugging tool only in the debugging mode, so such a debugging terminal is not used when in the user mode.

Since the input of a forced break is for switching from the user mode to debugging mode, it occurs only in the user mode.

Since the input of debugging information and the input of a forced break occur in different modes, they can be clearly differentiated even if the same terminal is used therefor, causing no confusion.

Since the present invention ensures that a terminal that is necessary during on-chip debugging and a terminal for inputting a forced break is used in common, it is possible to ensure that a larger number of terminals can be utilized by the user.

In a further aspect of the present invention, the microcomputer comprises: a first monitor means for transferring data to and from a second monitor means, determining a primitive command to be executed according to the data received from the second monitor means, and executing the determined primitive command, the second monitor means being provided outside the microcomputer for converting a debugging command into at least one primitive command; and the ~~debugging terminal connected to a single communications line for transferring the data in a half-duplex bidirectional manner~~,

the central processing unit executes a user program when in the user mode and executes the primitive command when in the debugging mode, and

the switching means switches the central processing unit from the user mode to the debugging mode when a forced break is input through the debugging terminal.

In this aspect of the invention, a second monitor means provided outside the microcomputer performs processing to convert (parse) debugging commands that have been developed by a host system or the like, into primitive commands. A first monitor means receives data from this second monitor means and executes primitive commands that are determined according to this received data. With this aspect of the invention, it is no longer necessary to ensure that a monitoring program for executing the processing of the first monitor means has complicated routines for executing debugging commands. This makes it to greatly reduce the instruction code size of the monitor program, thus implementing an on-chip debugging function that has a small hardware scale.

The number of debugging terminals of the microcomputer can

also be reduced, which tends to reduce the cost of the microcomputer.

Yet another aspect of the present invention further comprises means for holding a terminal for the input of a forced break at a first level which is either one of high and low, during a state in which no external debugging tool is connected, wherein the central processing unit starts execution in the user mode when the terminal for inputting the forced break is at the first level at a time of reset, or starts execution in the debugging mode when the terminal for inputting the forced break is not at the first level at a time of reset.

In this aspect of the invention, the terminal is held at the first level that is either high or low when no external debugging tool is connected. Therefore, if the configuration is such that the level of the terminal for the input of a forced break goes to the opposite level from the first level, by connecting this debugging terminal, it is possible to determine whether operation is to start in the user mode or the debugging mode, from the state of the terminal at a time of reset.

Since the mode is to be debugging when an external debugging tool is connected, it is preferable that execution starts in the debugging mode at a time of reset in such a case. If no debugging tool is connected, it is preferable that execution starts in the user mode, without the user being aware of any difference.

This aspect of the invention has a simple configuration that detects whether the debugging terminal is high or low, so that execution can start in the appropriate mode at a time of reset, without the user being aware of any difference.

Electronic equipment in accordance with another aspect of the

present invention comprises any of the above described microcomputers; an input source of data that is to be a processing object of the microcomputer; and an output device for outputting data that has been processed by the microcomputer.

This makes it possible to be more efficient in the debugging operations such as programs for operating the electronic equipment, shortening the development time of the electronic equipment and reducing the cost thereof.

A final aspect of the present invention relates to a debugging system for a target system including a microcomputer, the debugging system comprising: a second monitor means for performing processing for converting a debugging command developed by a host system into at least one primitive command; a first monitor means for transferring data to and from the second monitor means, determining a primitive command to be executed according to the data received from the second monitor means, and executing the determined primitive command; a central processing unit formed to be switchable between a user mode and a debugging mode, for executing the primitive command in the user mode; a debugging terminal provided on a chip including the central processing unit and connected to a single communications line for transferring the data in a half-duplex bidirectional manner; and switching means for switching the central processing unit from the user mode to the debugging mode when a forced break is input through the debugging terminal.

With this aspect of the invention, the instruction code size of a monitor program for executing the processing of the first monitor means can be greatly reduced. This makes it possible to increase the number of terminals and the memory region that can be

freely utilized by the user. In addition, it is also possible to provide a debugging system that enables debugging in an environment that is the same as that of the target system in actual operation.

Brief Description of Drawings

Fig. 1 shows an example of a CPU-switching ICE;

Figs. 2A and 2B illustrate a characteristic of the present invention;

Fig. 3 is a timing chart of the relationship between the input of a forced break and the state of the SIOD terminal;

Figs. 4A and 4B illustrate a second characteristic of the present invention;

Figs. 5A and 5B illustrate the relationship between the state of the SIOD terminal and the start mode at reset;

Fig. 6 illustrates a third characteristic of the present invention;

Fig. 7 is a functional block diagram of an example of the structure of the microcomputer and debugging system of the present embodiment;

Fig. 8 shows the memory map in debugging mode;

Figs. 9A to 9D illustrate the processing involved in the conversion (parsing) of debugging commands into primitive commands;

Fig. 10 is a functional block diagram of an example of the structure of the SIO;

Fig. 11 is a functional block diagram of an example of the structure of the debugging tool;

Figs. 12A to 12C illustrate communication methods between the mini monitor section and the main monitor section;

Fig. 13 illustrates the transition from user program run mode to debugging mode;

Fig. 14 is a flowchart of details of the processing of the present embodiment;

Fig. 15 is another flowchart of details of the processing of the present embodiment;

Figs. 16A to 16C show internal block diagram of various items of electronic equipment; and

Figs. 17A to 17C shows external views of the electronic equipment.

Best Mode for Carrying Out the Invention

Preferred embodiments of the present invention are described below with reference to the accompanying drawings.

1. Characteristics of the present embodiment

Characteristics of the present embodiment will first be described with reference to Figs. 2A and 2B. Each of Figs. 2A and 2B shows a microcomputer 10 having a user mode and a debugging mode, connected to a debugging tool 14.

The microcomputer 10 comprises a central processing unit (CPU) 12, an on-chip debugging section 13, a forced break control section 15, and a clock generation section 17. The on-chip debugging section 13 transfers debugging information to and from the debugging tool 14 when in a debugging mode, and executes a debugging program to perform various types of debugging.

The microcomputer 10 has an SIOD 16 and a BCLK 18 as debugging terminals. The SIOD 16 accepts the input of a forced break signal

when in user mode, and transfers debugging information to and from the debugging tool 14 by start-stop synchronization when in debugging mode. The BCLK 18 is supplied from the clock generation section 17 and is used as a synchronization clock for start-stop synchronization.

Incidentally, communication with the external debugging tool 14 is done only in debugging mode; these debugging terminals are not used when in user mode. On the other hand, the input of a forced break occurs only in user mode, for input for switching from user mode to debugging mode.

Therefore, the forced break control section 15 judges that a forced break has been input when a signal is received in user mode, and performs processing to switch the CPU 12 from user mode to debugging mode (see Fig. 2A). When such a signal is received in debugging mode, it is judged to be on-chip debugging information and that debugging information is output to the on-chip debugging section 13 (see Fig. 2B).

Since the transfer of debugging information and the input of a forced break occur only in different modes, there is no confusion even although the same terminal is used in common therefor, as shown in Figs. 2A and 2B.

In the present embodiment, the SIOD 16 is configured to function as a terminal for inputting a signal for a forced break when in user mode, as shown in Fig. 2A, and to function as a terminal for the transfer of debugging information in debugging mode, as shown in Fig. 2B.

Thus the number of terminals necessary during debugging can be reduced, making it possible to ensure that a larger number of

terminals can be utilized by the user.

A timing chart of the relationship between the input of a forced break and the state of the SIOD terminal is shown in Fig. 3.

Reference number 16' denotes the SIOD output state on the debugging tool 14 side and reference number 16 denotes the SIOD input state on the microcomputer side. A high-level signal is output from the debugging tool side when in user mode, but a low-level pulse is output by the input of a forced break from the outside (222). This low-level pulse is received by the microcomputer and user mode switch to debugging mode (228).

After the debugging tool 14 side has output a high-level signal (224) for a constant-period, the communication of debugging information starts under start-stop synchronization.

A second characteristic of the present embodiment will now be described with reference to Figs. 4A and 4B. When no debugging tool is connected to the microcomputer 10, the SIOD terminal is kept at high to pull it up, as shown in Fig. 4A. However, the SIOD terminal 16 can be set to any level (either high or low) by connecting the debugging tool 14 as shown in Fig. 4B.

In the present embodiment, a low-level signal is output from the debugging tool 14 to force the SIOD terminal 16 low, by connecting the debugging tool 14 to the microcomputer 10.

The relationship between the state of the SIOD terminal and the initial mode at reset will now be described with reference to Figs. 5A and 5B.

When the SIOD is high at the rise of a user RESET signal (230), as shown in Fig. 5A, the microcomputer starts operating in user mode (232).

When the SIOD is low at the rise of the user RESET signal (234), as shown in Fig. 5B, the microcomputer starts operating in debugging mode (236). Therefore, the microcomputer shifts to debugging mode (238) and the debugging tool side outputs a high-level signal (240) for a constant-period, then starts communicating debugging information under start-stop synchronization (242).

It is therefore possible to start execution in the appropriate mode at reset, without the user being aware of any difference, with a simple configuration that detects whether the SIOD is high or low at user reset.

A third characteristic of the present embodiment of the invention will now be described. This case is equivalent to the on-chip debugging section 13 of Figs. 2A and 2B in case that it has a function equivalent to a mini monitor section 314 which will be described below.

As shown in Fig. 6, the microcomputer 10 of the present embodiment comprises the central processing unit (CPU) 12 and a mini monitor section (first monitor means) 314. In addition, a main monitor section (second monitor means) 316 is provided outside the microcomputer 10. The main monitor section 316 performs processing to convert (parse) debugging commands developed by a host system, for example, into primitive commands. In addition, the mini monitor section 314 transfers data to and from the main monitor section 316. The mini monitor section 314 determines the primitive commands to be executed, based on the data received from the main monitor section 316, and performs processing for executing those primitive commands.

In this case, commands such as program load, GO, step execution,

memory write, memory read, internal register write, internal register read, breakpoint setting, or breakpoint release could be considered as the debugging commands that are the object of the conversion processing performed by the main monitor section 316. The main monitor section 316 executes processing to convert diverse, complicated debugging commands, such as GO, write (a write to a given address on the memory map, when in debugging mode), and read (a read from a given address on the memory map), into simple primitive commands. Such a configuration makes it possible to greatly reduce the instruction code size of the mini monitor program run by the mini monitor section 314. This enables the execution of an on-chip debugging function for the microcomputer 10.

A conventional debugging program comprises all the processing routines for debugging commands, such as program load, GO, and step execution, making it necessary to provide a large memory therefor and thus making it difficult to install in a microcomputer.

However, the mini monitor program run by the mini monitor section 314 of the present invention only has processing routines containing simple primitive command such as GO, write, and read, making the instruction code size thereof extremely small (256 bytes, for example). This means that the mini monitor program can be installed in the microcomputer 10, enabling the implementation of an on-chip debugging function. Further, any reduction of the memory region that can be used freely by the user, can be restrained to minimum or even to zero.

2. Detailed Structural Example

A detailed example of the structure of the microcomputer and

debugging system of the present embodiment is shown in Fig. 7. As shown in Fig. 7, a microcomputer 20 comprises a CPU 22, a bus control unit (BCU) 26, internal memory (internal ROM and internal RAM other than a mini monitor ROM 42 and a mini monitor RAM 44) 28, a clock generation section 30, a mini monitor section 40 (first monitor means), and a trace section 50.

In this case, the CPU 22 executes various instructions and comprises internal registers 24. The internal registers 24 comprise general-purpose registers R0 to R15 as well as a stack pointer (SP) register, a higher arithmetic register (AHR) for storing sum-of-products result data, and a lower arithmetic register (ALR) for storing sum-of-products result data, which are special registers. Note that the CPU 22 has a user mode and a debugging mode and is configured to switch from user mode to debugging mode on the input of a forced break from a forced break control section 49 through a line 51.

A BCU 26 controls buses. For example, it controls a bus 31 of a Harvard architecture connected to the CPU 22, a bus 32 connected to internal memory 28, an external bus 33 connected to external memory 36, and an internal bus 34 connected to components such as the mini monitor section 40 and the trace section 50. The clock generation section 30 generates the various clock signals used within the microcomputer 20. The clock generation section 30 also supplies a clock signal to the debugging tool 60 through the BCLK.

The mini monitor section 40 comprises the mini monitor ROM 42, the mini monitor RAM 44, a control register 46, an SIO 48, and the forced break control section 49.

In this case, a mini monitor program is stored in the mini

monitor ROM 42. The mini monitor program in the present embodiment executes only simple primitive commands such as GO, read, and write. Thus the memory capacity of the mini monitor ROM 42 can be restrained to about 256 bytes, by way of example, and thus the microcomputer 20 can be made more compact while still retaining an on-chip debugging function.

The contents of the internal registers 24 of the CPU 22 are saved to the mini monitor RAM 44 at a transition to debugging mode (when a break occurs in a user program). This ensures that the execution of the user program can restart correctly after debugging mode ends. Reading and other manipulation of the contents of these internal registers can be implemented by primitive commands within the mini monitor program, such as a read command.

The control register 46 is a register for controlling the various debugging processes, and contains a step execution enable bit, a break enable bit, a break address bit, and a trace enable bit, and the like. The CPU 22 operating in accordance with the mini monitor program can implement the various debugging processes by writing data to the bits of the control register 46 and reading data from those bits.

The SIO 48 controls debugging data that is transferred to and from the debugging tool 60 that is provided outside the microcomputer 20.

The forced break control section 49 and the debugging tool 60 are connected by the SIOD (data transfer line) which inputs a forced break and transfers debugging information.

The forced break control section 49 judges that a forced break has been input, on receiving a signal through the SIOD in user mode,

a

and sends a signal through the line 51 for switching the ~~CPU 12~~ CPU 22 from user mode to debugging mode. In case it receives a signal when in debugging mode, it judges it to be debugging information and perform processing to output that debugging information to the SIO 48.

The trace section 50 implements a real-time trace function. The trace section 50 and the debugging tool 60 are connected by four lines: a 3-bit DST [2:0] indicating the state of instruction execution at the CPU 22 and a DPCO indicating the program counter (PC) of the branch destination.

The debugging tool 60 comprises a main monitor section 62 and is connected to a host system 66 implemented by a personal computer or the like. The host system 66 issues debugging commands such as program load and step execution in answer to the user's operation, and the main monitor section 62 converts (parses) those debugging commands into primitive commands. When the main monitor section 62 sends data directing the execution of primitive commands to the mini monitor section 40, the mini monitor section 40 executes the directed primitive commands.

An example of the memory map in debugging mode is shown in Fig. 8. The addresses of the control register 46, the mini monitor RAM 44, and the mini monitor ROM 42, shown in Fig. 7, are allocated on the memory map when in debugging mode, as shown at D1, D2, and D3 in Fig. 8.

3. Conversion to Primitive Commands

The conversion of various debugging commands into primitive commands is shown schematically in Figs. 9A to 9D.

Assume, by way of example, that a debugging command is issued

to load 12-byte program data (ADD..., SUB..., AND..., OR..., XOR..., LD.W...) to an address 80010h, as shown in Fig. 9A. In this case, this program load command is converted into three primitive write commands: write (80010h, ADD..., SUB), write (80014h, AND..., OR...), and write (80018h, XOR..., LD.W...). In other words, the mini monitor program implements a program load command by executing these three primitive write commands.

Assume that a debugging command that is a step execution command is issued, as shown in Fig. 9B. When this happens, this step execution command is converted into a write command with respect to the step execution enable bit of the control register 46 of Fig. 7 (a write command to the address at D1 in Fig. 8) and a GO command. In other words, the mini monitor program executes these primitive write and GO commands so that the step execution command is implemented.

Assume that a debugging command that is an internal register read command is issued, as shown in Fig. 9C. When this happens, this internal register read command is converted into a read command from the mini monitor RAM (the save destination of the contents of the internal registers) on the memory map (a read command from the address at D2 in Fig. 8). In other words, the mini monitor program executes this primitive read command so that an internal register read command is implemented. An internal register write command, a memory read command, and a memory write command are all implemented in a similar fashion.

Finally, assume that a debugging command that is a breakpoint setting command is issued, as shown in Fig. 9D. When this happens, this breakpoint setting command is converted into write commands

to the break enable bit and break address bit of the control register 46. In other words, the mini monitor program executes these primitive write commands so that a breakpoint setting command is implemented.

In the thus-configured embodiment, complicated, diverse debugging commands can be converted into simple and primitive read, write, and GO commands. In addition, the instruction code size of the mini monitor program is extremely small, because only these primitive read, write, and GO commands need be executed. As a result, the memory capacity of the mini monitor ROM 42 can be made small and an on-chip debugging function can be implemented within a compact hardware structure.

4. Structural Example of SIO

An example of the structure of the SIO 48 is shown in Fig. 10. The SIO 48 comprises a transmission buffer 70, a shift register 76, a send/receive switching section 78, a clock control section 80, and a control register 84.

In this case, the transmission buffer 70 holds send data and receive data temporarily, and comprises a send buffer 72 and a receive buffer 74. The shift register 76 has the functions of converting send data from the send buffer 72 from parallel data into serial data, then outputting it to the send/receive switching section 78. It also has the functions of converting receive data from the send/receive switching section 78 from serial data into parallel data, then outputting it to the receive buffer 74. The send/receive switching section 78 switches between sending and receiving data. This enables half-duplex data transfer, using the SIOD.

The clock control section 80 uses an incorporated clock division circuit 82 to divide BCLK and output a sampling clock SMC1 obtained by this division to the shift register 76. The operation of the shift register 76 is based on this SMC1. Since the BCLK is also supplied to the debugging tool 60, the BCLK can be used in common by the microcomputer 20 and the debugging tool 60.

The division ratio of the clock division circuit 82 is set by the control register 84. In other words, the mini monitor program executed by the CPU 22 can set the division ratio of the clock division circuit 82 by writing a predetermined division ratio to the control register 84.

5. Structural Example of Debugging Tool

An example of the structure of the debugging tool 60 is shown in Fig. 11.

A CPU 90 executes a program stored in a ROM 108, providing overall control of the debugging tool 60. A send/receive switching section 92 switches between the transmission and reception of data. A clock control section 94 controls a clock signal supplied to the SCLK terminals of the CPU 90, an address incrementer 100, and a trace memory 104. The BCLK from the microcomputer 20 (the SIO 48) is input to this clock control section 94. The clock control section 94 comprises a frequency detection circuit 95 and a clock division circuit 96. The frequency detection circuit 95 detects the frequency range to which the BCLK belongs, then outputs the result to a control register 98. In addition, the division ratio of the clock division circuit 96 is controlled by the control register 98. In other words, a main monitor program executed by the CPU 90 (stored in a main

monitor ROM 110) reads out the frequency range of the BCLK from the control register 98. The main monitor program determines the optimal division ratio corresponding to this frequency range, and writes that division ratio to the control register 98. The clock division circuit 96 divides BCLK by this division ratio to generate SMC2, which it outputs to the SCLK terminal of the CPU 90.

The address incrementer 100 increments the address in, the trace memory. A selector 102 selects either one of a line 122 (the address output by the address incrementer 100) or a line 124 (an address from an address bus 120), to output data to an address terminal of the trace memory 104. Another selector 106 selects either one of a line 126 (DST [2:0] and DPCO, which are output by the trace section 50 of Fig. 3) or a line 128 (a data bus 118), to output data to a data terminal of the trace memory 104 or extract data from that data terminal.

The ROM 108 comprises the main monitor ROM 110 (equivalent to the main monitor section 62 of Fig. 3), and a main monitor program is stored in the main monitor ROM 110. This main monitor program performs processing for converting debugging commands into primitive commands, as described previously with respect to Figs. 5A to 5D. A RAM 112 acts as a work area for the CPU 90.

An RS232C interface 114 and a parallel interface 116 function as interfaces to the host system 66 of Fig. 7, so that debugging commands from the host system 66 are input to the CPU 90 through these interfaces. ~~A clock generation section 118~~ A Clock generation section 119 generates the clock that activates the CPU 90.

6. Data Transfer

A method by which TXD (transmission) and RXD (reception) lines are separately provided and communication is full-duplex could be considered for the communication of debugging data between the mini monitor section 40 and the main monitor section 62, as shown in Fig. 12A.

However, note that if two lines (terminals) are used for communication of this debugging data, the number of terminals (number of pins) of the microcomputer would be increased thereby, leading to an increase in the cost of the microcomputer.

In the present embodiment, a single TXD/RXD line (bidirectional communications line) is provided between the mini monitor section 40 and the main monitor section 62, as shown in Fig. 12B, and half-duplex bidirectional communication is performed. Note that, in the present embodiment, this line is also used as the SIOD for the input of a forced break. This means that the increase in the number of terminals of the microcomputer can be restrained to a minimum, lowering the cost of the microcomputer.

Furthermore, as shown in Fig 12C, when the condition is such that the mini monitor section 40 that is acting as a slave has received data from the main monitor section 62 that is acting as master, it performs processing corresponding to that receive data and sends response data in answer to that receive data back to the main monitor section 62. In other words, when the main monitor section 62 sends data (commands) to the mini monitor section 40, the mini monitor section 40, which is in a wait state, receives the data and performs processing in accordance with the thus received data. The data (reply) in response to the receive data is sent to the main monitor section 62. Subsequently, the mini monitor section

40 goes into the wait state until data is again received from the main monitor section 62. In other words, the operation of the mini monitor section 40 is halted until data is received from the main monitor section 62, and operation starts on the condition that data has been received. This configuration makes it possible to transfer data in a suitable manner between the mini monitor section 40 and the main monitor section 62, which using a single communications line.

7. Detailed Processing Example of Mini Monitor Section

The description now turns to a detailed example of the processing of the mini monitor section.

If a break is generated while a user program is running, the processing of the mini monitor program starts and a CPU changes from user program execution mode to debugging mode, as shown in Fig. 13. When the mini monitor program processes a given command and executes a return instruction, the CPU returns from debugging mode to user program run mode.

Flowcharts of the processing of the mini monitor program in debugging mode are shown in Figs. 14 and 15.

When debugging mode is activated, the mini monitor program first saves the contents of the internal registers 24 of the CPU 22 of Fig. 7 in ~~the monitor RAM 44~~ the MINI MONITOR RAM 44 (step S1). The control register 46 used by the mini monitor program is then set (step S2).

14-byte data received from the debugging tool 60 is written to the receive buffer 74 of Fig. 10 (step S3). The first one byte of data in the receive buffer 74 (command identification data ID) is checked (step S4).

If the ID indicates a read command, as shown in Fig. 13, a read address is fetched from the receive buffer 74 (steps S5 and S6). Data is then read from the thus-fetched read address, and is written to the send buffer 72 (step S7). The data in the send buffer 72 is then sent to the debugging tool 60 (step S8). The processing returns to step S3 of Fig. 14 and the next receive data is written to the receive buffer 74.

If the ID indicates a write command, a write address is fetched from the receive buffer 74 (steps S9 and S10). Write data is then fetched from the receive buffer 74 and is written to the write address obtained in step S10 (step S11).

If the ID indicates an external routine jump command, the routine's address is fetched from the receive buffer 74 (steps S12 and S13). After the jump to the external routine, processing returns to the mini monitor program (step S14).

If the ID indicates a GO command, the data saved to the mini monitor RAM 44 is restored to the internal registers 24 (steps S15 and S16). Control then returns to the user program shown in Fig. 13 and debugging mode is canceled (step S17).

If the ID indicates that this is neither a read, write, external routine jump, nor GO command, on the other hand, the system judges that processing is not necessary (steps S15 and S18). Dummy data is then written to the send buffer 72 (step S19). Note that the processing of data fill commands is omitted from Fig. 15.

As described above, the configuration is such that primitive commands obtained by converting debugging commands are executed by the mini monitor program.

8. Electronic Equipment

The description now turns to electronic equipment comprising the microcomputer of the present embodiment.

An internal block diagram of a car navigation system that is one example of such electronic equipment is shown in Fig. 16A and an external view thereof is shown in Fig. 17A. A remote controller 510 is used to operate this car navigation system and the position of the vehicle is detected by a position detection section 520 based on information from GPS or gyroscope. Maps and other information are stored in a CD-ROM 530 (information storage medium). An image memory 540 functions as a work area during image processing, and the thus generated images are displayed to the driver by an image output section 550. A microcomputer 500 inputs data from data input sources such as the remote controller 510, the position detection section 520, and the CD-ROM 530, performs various operations thereon, then uses an output device such as the image output section 550 to output the data after the processing.

An internal block diagram of a game machine that is another example of such electronic equipment is shown in Fig. 16B and an external view thereof is shown in Fig. 17B. Using an image memory 590 as a work area, this game machine generates game images and sounds based on the player's operating information from a game controller 560, a game program from a CD-ROM 570, and player information from an IC card 580, and outputs them by using an image output section 610 and a sound output section 600.

An internal block diagram of a printer that is a further example of such electronic equipment is shown in Fig. 16C and an external view thereof is shown in Fig. 17C. Using a bit map memory 650 as

a work area, this printer generate print images based on operating information from an operating panel 620 and character information from a code memory 630 and font memory 640, and outputs them by using a print output section 660. A display panel 670 is used for conveying the current state and mode of the printer to the user.

The microcomputer or debugging system in accordance with the present embodiment makes it possible to simplify the development and reduce the development time of user programs that cause the operation of the items of electronic equipment shown in Figs. 16A to 17C. Since it also makes it possible to debug user programs in an environment that is the same as that in which the microcomputer operates, the reliability of this electronic equipment can also be increased. The hardware of the microcomputer installed into this electronic equipment can be made more compact and less expensive, leading to a reduction of the cost of the electronic equipment itself. Since the instruction code size of the mini monitor program is also small, the memory area used by the user for storing programs and various data is completely untouched thereby.

Note that the electronic equipment to which the microcomputer of the present embodiment can be applied is not limited to those described in the above examples, and thus it could be any of a portable telephone (cellular phone), a PHS, a pager, audio equipment, an electronic organizer, an electronic tabletop calculator, a POS terminal, a device provided with a touch panel, a projector, a dedicated wordprocessor, a personal computer, a television set, or a view-finder or direct monitor type of video tape recorder, by way of example.

Note also that the present invention is not limited to the

embodiments described herein, and various modifications can be conceived within the scope of the invention.